VIETNAM GENERAL CONFEDERATION OF LABOUR

**TON DUC THANG UNIVERSITY**

**FACULTY OF INFORMATION TECHNOLOGY**



**DESIGN PATTERN**

**VISITOR PATTERN AND STATE PATTERN**

*Supervisor*:  **NGUYEN THANH PHUOC**

*Author*: **NGO GIA PHAT – 519H0113**

**LE DAC CHINH – 519H0274**

**BAO HOANG QUI – 519H000**

Class: **19H50202 – 19H50302**

Group**: 2**

**HO CHI MINH CITY, 2023**

VIETNAM GENERAL CONFEDERATION OF LABOUR

**TON DUC THANG UNIVERSITY**

**FACULTY OF INFORMATION TECHNOLOGY**



**DESIGN PATTERN**

**VISITOR PATTERN AND STATE PATTERN**

*Supervisor*:  **NGUYEN THANH PHUOC**

*Author*: **NGO GIA PHAT – 519H0113**

**LE DAC CHINH – 519H0274**

**BAO HOANG QUI – 519H000**

Class: **19H50202 – 19H50302**

Group**: 2**

**HO CHI MINH CITY, 2023**

APPRECIATION LETTER

First and foremost, I would want to express my gratitude to Ton Duc Thang University for providing facilities that include a contemporary library system, a diverse collection of books and materials, and a conducive environment for obtaining and researching knowledge. Thank you very much! Then, I'd want to express my gratitude to one of my professors, Nguyen Thanh Phuoc, who patiently led and instructed me throughout the process of creating the final project.

There are many flaws and restrictions in my final project due to my limited knowledge and reasoning abilities. We look forward to the assistance and contributions of the instructors and teachers to make my final project more comprehensive. Thank you sincerely!

**PROJECT COMPLETED AT TON DUC THANG UNIVERSITY**

I hereby declare that this is the product of my/our own project and under the guidance of Dr. Nguyen Thanh Phuoc;. The research content, results in this topic are honest and not published in any form before. The data in the tables for analysis, comments and evaluation are collected by the author himself from different sources, which are clearly stated in the reference section.

In addition, the project also uses a number of comments, assessments as well as data from other authors, other agencies and organizations, with citations and source annotations.

**If I find any arduous problems, I would like to take full responsibility for the content of my project**. Ton Duc Thang University is not related to the rights and copyright crimes caused by me during the implementation process (if any).

*Ho Chi Minh city, day month year*

*Author*

*(sign and write full name)*

*Ngo Gia Phat*

*Le Dac Chinh*

*Bao Hoang Qui*

TEACHER'S CONFIRMATION AND ASSESSMENT SECTION

**The confirmation part of the instructor**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ho Chi Minh city, day month year

(sign and write full name)

**The evaluation part of the teacher marks the test**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ho Chi Minh city, day month year

(sign and write full name)

SUMMARIZE

Visitor Pattern:

The Visitor pattern is suitable for complex object structures that require an operation without modifying the objects themselves. This pattern is often used in compilers and interpreters where the abstract syntax tree represents the program being compiled or interpreted.

The Visitor pattern separates an algorithm from the objects it operates on. It involves defining a visitor class that encapsulates the behavior that needs to be added, and then implementing a visitor interface in the objects to be visited. The visitor class then traverses the object structure and calls the appropriate methods on the objects to perform the desired operation.

State Pattern:

The State pattern is useful in scenarios where an object's behavior depends on its internal state, and the number of states and transitions between them are complex. This pattern is often used in user interfaces where the behavior of a component depends on the state of other components or user inputs.

The State pattern involves defining a state interface and implementing state classes that define the behavior for each state. A context class holds a reference to the current state and delegates requests to the state objects. When the internal state of the context changes, it switches to a different state object that handles subsequent requests.

In summary, the Visitor and State patterns are powerful tools for addressing complex and dynamic scenarios in Java applications. By separating algorithms from objects in the Visitor pattern and handling state transitions in the State pattern, developers can create more flexible and maintainable code.

TABLE CONTENT

[APPRECIATION LETTER i](#_Toc131035932)

[TEACHER'S CONFIRMATION AND ASSESSMENT SECTION iii](#_Toc131035933)

[SUMMARIZE iv](#_Toc131035934)

[TABLE CONTENT 1](#_Toc131035935)

[LIST OF TABLES, PICTURES, GRAPHS 2](#_Toc131035936)

[CHAPTER 1 – VISITOR PATTERN 3](#_Toc131035937)

[1.1 What is Visitor Pattern? 3](#_Toc131035938)

[1.2 Problem 3](#_Toc131035939)

[1.3 Solution 3](#_Toc131035940)

[1.4 Structure 4](#_Toc131035941)

[1.5 Pseudocode 4](#_Toc131035942)

[1.6 Implementation 4](#_Toc131035943)

[1.6.1 Implement in Java 4](#_Toc131035944)

[1.6.2 Description 4](#_Toc131035945)

[1.7 Conclusion 4](#_Toc131035946)

[1.7.1 Applicability 4](#_Toc131035947)

[1.7.2 Pros and Cons 5](#_Toc131035948)

[1.7.3 Relations with Other Patterns 6](#_Toc131035949)

[CHAPTER 2 – ABC PATTERn 6](#_Toc131035950)

[2.1 Results question a 6](#_Toc131035951)

[CHAPTER 3 – REFERENCE 6](#_Toc131035952)

[3.1 Acronym 6](#_Toc131035953)

[3.2 Citation 7](#_Toc131035954)

[3.2.1 References and citations 7](#_Toc131035955)

[3.2.2 Regulations of the Faculty of Information Technology 7](#_Toc131035956)

LIST OF TABLES, PICTURES, GRAPHS

CHAPTER 1 – VISITOR PATTERN

* 1. What is Visitor Pattern?

The Visitor pattern is a design pattern that separates an algorithm from the objects it operates on. It is useful when you have a complex object structure that you want to perform some operation on, but you don't want to modify the objects themselves. The Visitor pattern provides a way to add new operations to existing object structures without changing those structures.

* 1. Problem

A company has developed a large and complex system for managing their customer data. The system includes a variety of different types of objects, such as customers, orders, and invoices. The company wants to be able to generate reports that analyze different aspects of the customer data, such as total sales by region or average order size by customer type. However, adding new reporting functionality to the system is difficult because the objects in the system are not designed to support the required operations.

* 1. Solution

To solve this problem, the company decides to use the Visitor pattern. They create a ReportVisitor class that defines the behavior for generating reports, such as calculating total sales or computing averages. They then implement a Visitor interface in the various objects in the system that need to be analyzed, such as the customer, order, and invoice classes. The Visitor interface includes methods for accepting the ReportVisitor and delegating the required calculations to the visitor.

When the company needs to generate a report, they create a new instance of the ReportVisitor class and pass it to the root object in the system. The ReportVisitor traverses the object structure, visiting each relevant object and performing the required calculations. The resulting report can then be displayed to the user or exported to a file.

By using the Visitor pattern, the company is able to add new reporting functionality to their system without modifying the existing object structure. They can easily create new ReportVisitor classes to analyze the data in different ways, making the system more flexible and adaptable to changing business needs.

* 1. Structure
  2. Pseudocode
  3. Implementation
     1. Implement in Java
     2. Description

In this example, the `ReportVisitor` interface defines the methods for visiting the **`Customer`**, **`Order`**, and **`Invoice`** objects, and the concrete implementation of `SalesReportVisitor` calculates the total sales for each object. The **`accept`** method in each object allows it to be visited by the **`ReportVisitor`**, and the **`SalesReportVisitor`** instance is created and used to visit the objects and calculate the total sales.

* 1. Conclusion
     1. Applicability

The Visitor pattern is useful in situations where a complex data structure (such as an object hierarchy) needs to be traversed and processed by multiple unrelated algorithms or operations.

Here are some specific situations where the Visitor pattern can be applied:

* Adding new operations to existing classes without modifying those classes: If a data structure has a large number of classes, adding new operations to all of them can be time-consuming and error-prone. With the Visitor pattern, new operations can be added by creating a new visitor class that implements the necessary behavior for each class.
* Separating algorithms from the objects on which they operate: The Visitor pattern allows algorithms to be defined in separate visitor classes, which can be applied to objects without modifying those objects. This can make the code more modular and easier to maintain.
* Maintaining open-ended sets of operations on a complex object structure: The Visitor pattern allows new visitor classes to be created without modifying existing classes, making it easy to add new operations to a complex object structure.
* Reducing code duplication: If multiple operations require similar logic for visiting an object structure, the Visitor pattern can help reduce code duplication by centralizing the visiting logic in a single place.

Overall, the Visitor pattern is a useful tool for dealing with complex object structures where multiple algorithms or operations need to be applied in a flexible and extensible way.

* + 1. Pros and Cons

Pros of the Visitor pattern include:

* **Separation of concerns:** The Visitor pattern separates algorithms from the objects on which they operate, which can make code more modular and easier to maintain.
* **Extensibility:** The Visitor pattern allows new operations to be added to an object structure without modifying existing classes, which can be useful in large and complex systems.
* **Open/Closed Principle:** The Visitor pattern supports the Open/Closed Principle, which states that software entities should be open for extension but closed for modification. The Visitor pattern allows new behavior to be added without modifying existing code.
* **Code reuse:** The Visitor pattern can help reduce code duplication by centralizing visiting logic in a single place.

Cons of the Visitor pattern include:

* **Complexity:** The Visitor pattern can add complexity to code by introducing new classes and interfaces.
* **Increased code size:** Implementing the Visitor pattern can result in increased code size, which can make the code harder to read and understand.
* **Dependency injection:** In some cases, the Visitor pattern requires dependency injection to work properly, which can add additional complexity to code.
* **Performance overhead:** The Visitor pattern can result in additional method calls and object allocations, which can impact performance in certain situations.

Overall, the Visitor pattern can be a powerful tool for dealing with complex object structures, but it should be used judiciously and with an eye towards balancing the benefits of increased flexibility and extensibility against the potential costs of increased complexity and performance overhead.

* + 1. Relations with Other Patterns

The Visitor pattern can be used in combination with other design patterns to solve complex problems. Some examples of patterns that can be used in conjunction with the Visitor pattern include:

* **Composite pattern:** The Visitor pattern can be used to traverse a composite object structure and perform operations on the individual components.
* **Iterator pattern:** The Visitor pattern can be used to implement iteration over an object structure, allowing the Visitor to visit each element in turn.
* **Strategy pattern:** The Visitor pattern can be used to implement different algorithms that can be applied to an object structure, allowing the Visitor to choose the appropriate algorithm at runtime.
* **Interpreter pattern:** The Visitor pattern can be used to interpret a language or expression by visiting the nodes in an abstract syntax tree.
* **Template Method pattern:** The Visitor pattern can be used as part of a template method, where the Visitor provides a concrete implementation of an abstract method that is called from the template method.

CHAPTER 2 – ABC PATTERn

1. Results question a

CHAPTER 3 – REFERENCE

1. Acronym

The abbreviation function cannot be used. Only abbreviate words, phrases or terms that are used repeatedly in the essay. Do not abbreviate long phrases, clauses, or infrequent phrases. If it is necessary to abbreviate words, terms, names of agencies and organizations, etc., they shall be abbreviated after the first writing, accompanied by abbreviations in the citation application. If there are too many abbreviations, there should be a list of abbreviations (in order A, B, C) at the beginning of the essay.

1. Citation
2. References and citations
3. Regulations of the Faculty of Information Technology

**-Plagiarism** is the use of words or ideas of others as their own in academic activities in particular and in creative activities in general. At Hoa Sen University, the following acts are considered plagiarism:

• Copy verbatim 02 (two) sentences in a row without quoting the source in accordance with regulations;

• Copying 03 (three) non-consecutive sentences verbatim without quoting the correct source;

• Rephrase or translate (translate) an entire idea of someone else without citing the correct source;

• Using more than 30% of the content of a final report written by oneself to submit to 2 different classes (same semester or different semester) without the consent of the lecturer;

• Copy part or all of someone else's work.

- When the thesis, project, large assignment is graded, if it is detected plagiarism, it will be immediately scored 0. Students will continue to be disciplined according to the regulations of the University.

**REFERENCES**

**APPENDIX**

This section includes the necessary content to illustrate or support the thesis content such as figures, forms, pictures. . . . if responses to a questionnaire are used, this sample questionnaire must be included in the Appendix in its original form used for the survey and poll; may not be summarized or modified. The sample calculations presented in summary in the forms should also be mentioned in the Appendix of the thesis. Appendices should not be thicker than the main body of the thesis